Amendments to the Specification:

Replace the title in its entirety with the following title show below.

"METHOD AND APPARATUS FOR NESTED CONTROL FLOW OF INSTRUCTIONS USING CONTEXT INFORMATION AND INSTRUCTIONS HAVING EXTRA BITS"

Replace paragraphs [0013]–[0014], [0020], [0023], and [0025]–[0028] in their entireties with the paragraphs shown below.

[0013] The method and apparatus further includes a first memory device storing a plurality of instructions wherein each of the plurality of instructions includes a plurality of extra bits, wherein the processor is operative to execute the plurality of instructions. The memory may be, but not limited to, a single memory, a plurality of memory locations, shared memory, CD, DVD, ROM, RAM, EEPROM, optical storage, microcode or any other non-volatile storage capable of storing digital data for use by the processor. Moreover, the plurality of instructions may be any suitable coded instructions encoded within any suitable programming language or other instructional operation.

[0014] The method and apparatus for nested control flow further includes a second memory device operably coupled to the processor, the memory device ~~receiving~~ receives and ~~incrementing~~ increments a counter ~~instruction~~ value upon the execution of one of the plurality of instructions. The method and apparatus, by the inclusion of the plurality of extra bits in conjunction with the context bit, provide for an improved nested control flow operation through the determination of whether to execute an instruction based not only on the plurality of extra bits, but also the value of the context bit itself.

[0020] Step 126 is determining whether to read the context bit based on the plurality of extra bits. This step is performed, in one embodiment, by the processor 102 based on examining the status of the extra bits, such as a true or false state or a yes or no state. Furthermore, if the

context bit 108 is read, the context bit is extracted from the memory location storing the context ~~bits~~ bit 108 such [[as]] that the processor 102 may read the context bit 108. Step 128 is executing the instruction when the context bit is in the first state, upon the reading of the context bit. As discussed above, the instruction is performed by the processor 102 which is a SIMD processor processing the single instruction upon multiple data sets. Thereupon, the method is complete, step 130.

[0023] The present invention utilizes a single context bit per processing element 152, 154 and 156. With the addition of two added bits to each instruction, an instruction can execute independent of the context bit 158, 160 and 162 or can check the context bit 158, 160 and 162 and execute only when the context bit 158, 160 and 162 is set to execute, such as an on position. The present invention eliminates the need for a per element counter by using a general purpose register to hold the counter, such as illustrated as the second memory device 106 of FIG. 1. Consequently, any ALU 152, 154 or 156 operations can be used to modify the counter. The general purpose register used for the counter can be non-dedicated and the same register does not need to be utilized for different kinds of execution of an overall program.

[0025] As recognized by one having ordinary skill in the art, these control flow sequences can be implemented using any combination of instruction from standard arithmetic operations. Using an exemplary program to illustrate operation of the present invention, an example ~~if then~~ if-then statement is the operation of:

If x is greater than 0 then

Y = 3

Y = [[u+v]] u + v

[0026] With respect to FIG. 3, representative storage locations within the general purpose registers 164, 166 and 168 have been designated as memory locations 180–184, 186–190 and 192–196 respectively. Based on the operation of the ALU 152, 154 and 156, specific data is written within the general purpose register memory locations 180–184, 186–190 and 192–196. A first operation is the register performing a predicate_push operation to determine if a value stored at Rx is greater than zero, such as looking at an initial value stored within a register for x and performing a predicate_push for register location 180, 186 and 192. Based on this comparison, a second register value may be computed as three (p) relative to register locations 181, 187 and 193. Thereupon, the computation of [[y=u+v]] y = u + v may be performed by the defining of the register value Ry as being the equivalent of the register value Ru in summation with the register value Rv (p). Therefore, register value 180, 186 and 192 would then be defined as a predicate_pop for the value within register locations 180, 186 and 192.

[0027] As recognized by one having ordinary skill in the art, the above example indicates one single nested operation within a control flow, wherein the present invention is utilized within multiple nested control flow operations. Therefore, in a nested control flow operation with multiple nested operations, the counter is implemented such that the number of nested operations into the depth of nested control flow may be effectively monitored and controlled when executing any nested flow operation. More specifically, ~~allow~~ this allows for a machine level instruction set for breaking out of an instruction and jumping around.

[0028] As discussed above, there exists any suitable implementation of conditional statements, although two common statements are an ~~if-then~~ if-then statement or a while statement. Included below in Table 2 are two representative examples of the implementation of

operations wherein the sequences can be ~~nested and other comment~~ nested; other common control flow sequences can be implemented as well.